Mapeamento Sistemático de Processos de Refatoração de Software

Pedro Magnus Pedroso Nogueira

pednog@alunos.utfpr.edu.br

Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

Profª. Drª. Simone Nasser Matos

snasser@utfpr.edu.br

Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná,

Recebido: 19 ago. 2019.

Aprovado: 01 out. 2019.

RESUMO

Este artigo relata um mapeamento sistemático para encontrar trabalhos relacionados a técnicas de refatoração de software e técnicas de detecção de *bad smells.* Dos trabalhos encontrados, foram selecionados 24 para análise. Estes trabalhos ajudaram a responder as questões criadas como guia para o mapeamento, as respostas foram respondidas após a leitura de todos os artigos e armazenado-os em um protocolo para ser consultado. Os resultados demonstram pontos importantes sobre a refatoração de software como a linguagem mais usada, autores mais influentes e ferramentas.

PALAVRAS-CHAVE: Refatoração de Software. Mapeamento Sistemática. Bad Smells.

ABSTRACT

This article reports a systematic mapping to find studies related to software refactoring techniques and bad smells detection techniques. From all the found studies, 24 were selected for analysis. These papers helped to answer the questions created as a guide for the mapping, the answers to the questions were answered after reading all the articles and stored them in a protocol to be consulted. The results demonstrate important points about software refactoring as the most used language, most influential authors and tools.

KEYWORDS: Software Refactoring . Systematic Mapping. Bad Smells.

INTRODUÇÃO

Refatoração é o processo de modificar um software sem alterar seu comportamento externo, visando somente alterar o código para otimizar a sua estrutura interna (FOWLER, 1999). Mesmo um sistema sendo projetado antes de ser programado, ocasionalmente problemas ou até mesmo aplicações erradas podem aparecer e esses "erros" são chamados de *bad smells*. A refatoração é aplicada para remover os *bad smells* fazendo com que o sistema tenha atributos de qualidade tais como flexibilidade, manutenibilidade e legibilidade.

Com o objetivo de conhecer o estado da arte sobre técnicas de refatoração de software, este artigo apresenta um mapeamento sistemático que foi realizado usando o método proposto por Kitchenham (2007), pois esse é o mais citado na área de engenharia de software. Este método permite planejar o processo de revisão identificando o problema em que é definido como o alvo do mapeamento sistemático, questões que devem definidas e respondidas ao final de sua aplicação.

MÉTODO DE PESQUISA

Para a criação de trabalhos com uma revisão da literatura confiável é necessário estabelecer critérios sistemáticos de busca e análise dos estudos. Por isto, a revisão ou mapeamento sistemático procuram estabelecer etapas de planejamento, execução e análise para detalhamento do estado da arte em um determinado assunto.

Após a realização de pesquisas o método de revisão ou mapeamento sistemático, constatou-se que Kitchenham (2007) é ideal para realização do estado da arte. O mapeamento busca analisar o conceito e como será realizada a pesquisa, procurando ter controle sobre os locais de pesquisas tal como as *keywords*. Isto permite que se tenha o resultado mais justo possível, sem possuir polarização e sempre com a melhor qualidade possível. O processo mais importante do protocolo é a formulação da pergunta para a pesquisa, com isso Kitchenham (2007) propõem seguir as *guidelines* criadas pela Australian NHMR (2000), mas com modificações feitas por ela para as *guidelines* se ajustarem na engenharia de software, já que elas foram inicialmente criadas para a medicina.

RESULTADOS

O principal objetivo da pesquisa é levantar técnicas de refatoração de software e as questões que foram identificadas são: *Q1. Qual o número de citação dos trabalhos?*, *Q2. A qual grupo de técnicas a refatoração pertence?*, *Q3.*

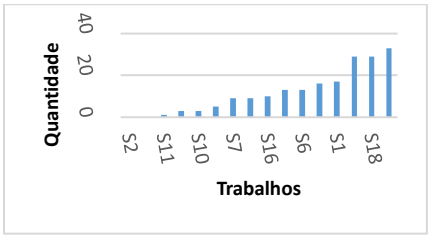
Existe alguma relação entre os trabalhos selecionados? e Q4. Quais ferramentas foram desenvolvidas?

Para encontrar os artigos a serem selecionados na revisão foram utilizadas repositórios de busca: Springer, ACM, SCIENCE DIRECT, WILEY, IEEEXplore e Google Scholar. A língua escolhida para a realização da pesquisa foi o Inglês. A string de busca foi composta com base nas palavras-chave. Após alguns testes, a string de busca ficou sendo “Refactoring technique” AND “Bad smell”.

Os seguintes critérios de inclusão de trabalhos foram adotados: possuir as palavras-chaves no título ou *abstract*; conteúdos relacionados com a refatoração de software por técnicas em seu *abstract* ou caso disponível introdução. Para a exclusão foram considerados os seguintes critérios: artigos duplicados; artigos que não foram disponibilizados pelos autores e artigos que não englobam os critérios de inclusão.

Após a escolha da string de pesquisa, todos os trabalhos passaram método Inclusão/Exclusão. Foram selecionados 24 trabalhos referentes ao assunto do mapeamento proposto para responder as questões. Em relação a *RQ1: Qual o número de citação dos trabalhos?* Os trabalhos que possuem citações menores que 50 totalizam 17 (dezessete) e estão exibidos no Gráfico 1. Os trabalhos receberam um código S_n e sua referência está listada na seção Estudos Selecionados.

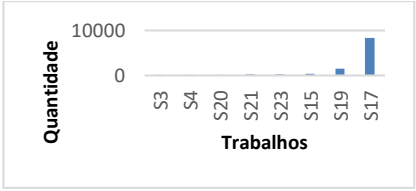
Gráfico 1 - Quantidade de citações menores que 50



Fonte: Autoria Própria (2019).

O Gráfico 2 exibe os demais trabalhos, em que as citações foram maiores do que cinquenta (50).

Gráfico 2 - Quantidade de citações maiores que 50

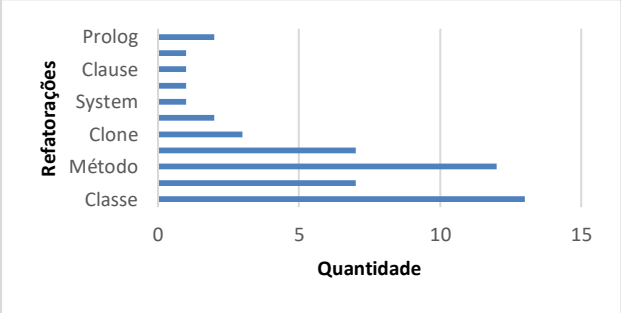


Fonte: Autoria Própria (2019).

Dois trabalhos possuem mais citações que os demais, S17 Fowler (1999) com 8.364 citações e S19 Opdyke (1992) com 1.568.

Em relação a *RQ2: A qual grupo de técnicas a refatoração pertence?* existem vários tipos de refatoração, como de: classe, campo e método. O resultado de quantidade de cada grupo está exibido no gráfico 3.

Gráfico 3 - Quantidade grupos de refatoração



Fonte: Autoria Própria (2019).

Em relação a *RQ3: Existe alguma relação entre os trabalhos selecionados?* Foram encontrados 11 trabalhos que focam suas refatorações em códigos escritos em Java, 4 em AspectJ, 2 em Prolog, 1em C++, 1 Smalltalk , 1 em .net, 1 em FLAT. Por fim, em relação a *RQ5: Quais ferramentas foram desenvolvidas? Foram* identificados 12 estudos que criaram ferramentas para refatoração.

CONCLUSÃO

Durante a realização do mapeamento foram percebidos alguns pontos em comum nos trabalhos, como sua linguagem de programação, suas técnicas de refatoração e se eles utilizam alguma ferramenta. A maior parte dos trabalhos utiliza a linguagem Java para refatoração com exceção dos trabalhos: S5, S6, S20, S24, S19, S12, S11, S15, S18. No trabalho S2 não é comentado a utilização de uma linguagem de programação para refatorar. Em relação as técnicas de refatoração a maioria dos estudos utiliza refatorações do tipo classe, interface, método e *field*. Em relação a ferramentas foi observado que metade dos trabalhos não possuem ferramentas e sim uma criação de uma técnica ou a melhoria de uma técnica já criada, mas em alguns desses trabalhos como S6 e S7 se propõem em criar uma ferramenta automática em trabalhos futuros.

ESTUDOS SELECIONADOS

S1 MKAOUER, M. W. et al. A robust multi-objective approach to balance severity and importance of refactoring opportunities. **Empirical Software Engineering**, v. 22, n. 2, p. 894–927, 4 abr. 2017.

S2 SANGEETHA, M.; CHANDRASEKAR, C. An empirical investigation into code smells rectifications through ADA_BOOSTER. **Ain Shams Engineering Journal**, fev. 2019.

S3 MOGHADAM, I. H.; CINNÉIDE, M. Ó. Automated Refactoring Using Design Differencing. In: EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, Szeged. **Proceedings**... Szeged (Hungary): IEEE, 2012.

S4 BAVOTA, G. et al. Automating extract class refactoring: an improved method and its evaluation. **Empirical Software Engineering**, v. 19, n. 6, p. 1617–1664, 4 dez. 2014.

S5 MONTEIRO, M, P. **Catalogue of Refactorings for AspectJ**. p. 50. 2004. Dissertação (Doutorado) - Escola Superior de Tecnologia de Castelo Branco Instituto Politécnico de Castelo Branco, Castelo Branco, 2004.

S6 UBAYASHI, N. et al. Contract-Based Verification for Aspect-Oriented Refactoring. In: INTERNATIONAL CONFERENCE ON SOFTWARE TESTING, VERIFICATION, AND VALIDATION, Lillehammer. **Proceedings**... Lillehammer (Norway): IEEE, abr. 2008.

S7 SRIVISUT, K. Definition and detection of bad smells of aspect-oriented program. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE. Beijing. **Proceedings**... Beijing (China): IEEE, 2007.

S8 RANI, A.; KAUR, H. Detection of bad smells in source code according to their object oriented metrics. **International Journal For Technological Research In Engineering**, June. 2014.

S9 RIESS, C. et al. EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In: INTERNATIONAL SEMANTIC WEB CONFERENCE, Shanghai. **Proceedings ...** Shangahai (China): Springer, Berlin, Heidelberg, 2010. p. 647–662.

S10 KAUR, S.; KAUR, H. **Identification and refactoring of bad smells to improve code quality.** 4 p. 2015. Dissertação - UCOE Punjabi University, Patiala, 2015.

S11 HUANG, J. et al. Identifying composite refactorings with a scripting language. In: INTERNATIONAL CONFERENCE ON COMMUNICATION SOFTWARE AND NETWORKS, Xi'an. **Proceedings**... Xi'an (China): IEEE, 2011, p. 572-577.

S12 SCHRIJVERS, T.; SEREBRENIK, A. Improving Prolog Programs: Refactoring for Prolog. In: INTERNATIONAL CONFERENCE ON LOGIC PROGRAMMING. Saint-Malo. **Proceedings**... Saint-Malo (France): Springer, Berlin, Heidelberg, 2004. p. 58–72.

S13 PANAMOOTTIL KRISHNAN, G. **Improving the Unification of Software Clones using Tree and Graph Matching Algorithms.** 79 p. Dissertação (Mestrado) - Concordia University, Québec, 2014.

S14 MKAOUER, M. W. et al. On the use of many quality attributes for software refactoring: a many-objective search-based software engineering approach. **Empirical Software Engineering**, v. 21, n. 6, p. 2503–2545, 23 dez. 2016.

S15 DONALD B ROBERTS. **Practical Analysis for Refactoring**. 127 p. 1999. Dissertação (Doutorado) - University of Illinois at Urbana-Champaign Champaign, IL, USA, 1999.

S16 ZHANG, M. et al. Prioritising Refactoring Using Code Bad Smells. In: INTERNATIONAL CONFERENCE ON SOFTWARE TESTING, VERIFICATION AND VALIDATION WORKSHOPS, Berlin. **Proceedings**... Berlin (Germany): IEEE, 2011.

S17 FOWLER, M.; BECK, K. **Refactoring :** improving the design of existing code. Addison-Wesley: Boston, 1999.

S18 CORREA, A.; WERNER, C. Refactoring object constraint language specifications. **Software & Systems Modeling**, v. 6, n. 2, p. 113–138, 4 jun. 2007.

S19 OPDYKE, W. **Refactoring object-oriented frameworks.** 206 p. Dissertação (Doutorado) - University of Illinois at Urbana-Champaign, Champaign 1992.

S20 HANENBERG, S**. Refactoring of aspect-oriented software.** 17 p. 2003.Dissertação - University of Duisburg-Essen. Essen, 2003.

S21 WAKE, W. C. **Refactoring workbook**. Addison-Wesley: Boston, 2004.

S22 WONGPIANG, R.; MUENCHAISRI, P. Selecting sequence of refactoring techniques usage for code changing using greedy algorithm. In: INTERNATIONAL CONFERENCE ON ELECTRONICS INFORMATION AND EMERGENCY COMMUNICATION, Beijing. **Proceedings**... Beijing (China)IEEE, 2013.

S23 MONTEIRO, M. P.; FERNANDES, J. M. **Towards a catalog of aspect-oriented refactorings**. In: INTERNATIONAL CONFERENCE ON ASPECT-ORIENTED SOFTWARE DEVELOPMENT, New York. **Proceedings**... New York, (USA): ACM Press, 2005, p. 111-122.

S24 SCHRIJVERS, T.; SEREBRENIK, A.; DEMOEN, B. **Refactoring Prolog code**. 12 p. Dissertação - K.U. Leuven, Celestijnenlaan, 2004.

REFERÊNCIAS

AUSTRALIAN NATIONAL HEALTH AND MEDICAL RESEARCH COUNCIL. **How to review the evidence: systematic identification and review of the scientific literature.** National Health and Medical Research Council: Australia, 2000.

CENTRE FOR REVIEWS AND DISSEMINATION. **Undertaking Systematic Review of Research on Effectiveness. CRD's Guidance for those Carrying Out or Commissioning Reviews.** Dissertação - University of York, 2001.

CHANDLER, J; et al. **Cochrane Reviewers' Handbook.** The Cochrane Collaboration, 2003.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007.

ZIMMER, J. A. Graph Theoretical Indicators and Refactoring. In: CONFERENCE ON EXTREME PROGRAMMING AND AGILE METHODS, New Orleans. **Proceedings**... New Orleans (USA): Springer, Berlin, Heidelberg, 2003. p. 62–72.